User Manual

CSE 453: Northrop Grumman Bird Chirp Classifier

Table of Contents

Setup and Installation

Ensure that the following libraries and packages are installed on a machine running Ubuntu 16.04 with a discrete NVIDIA GPU with CUDA Compute Capability 3.0:

1. **NVIDIA CUDA Toolkit v8.0**
    - This library is available for free from NVIDIA and can also be found under the following link: https://developer.nvidia.com/cuda-toolkit. Further instructions for installing this package are also provided by NVIDIA. This package is used to run code on a discrete NVIDIA GPU.
2. **libsndfile1-dev**
    - This package can be installed on Ubuntu by using the following command in a terminal: "sudo apt install libsndfile1-dev" without the quotes. This library is used by the feature extraction code to read from WAV files.
3. **cuDNN v5.1**
    - This library is available for free from NVIDIA and can also be found under the following link: https://developer.nvidia.com/cudnn. Further instructions for installing this package are also provided by NVIDIA.
4. **TensorFlow-GPU**
    - This library is available for free from TensorFlow and can also be found under the following link: https://www.tensorflow.org/install/install_linux. Further instructions for installing this package are also provided by TensorFlow.
    - This package can be installed on Ubuntu by using the following command in a terminal using native pip: "pip install tensorflow-gpu" without the quotes.
5. **Python v2.7**
    - Python is available for free and can be found under the following link: https://www.python.org/downloads/. Further instructions for installation can be found on their site.
6. **Scikit-learn**
    - Scikit-learn is available for free and can be found under the following link: http://scikit-learn.org/stable/install.html. Further instructions for installation can be found on their site.

Feature Extraction

The feature extraction is done using a program called extractFeatures, located in the CUDA/extractFeatures folder. This program is written in C++ and can be compiled using the accompanied Makefile which uses the NVCC compiler. There is a strict file naming convention to adhere to to ensure proper labelling of the output which is outlined in the **Filenames** section below. It is also worth noting that input files should be WAVE format files that are monophonic. Further information, such as how to make modifications to the program, can be found inside of the documentation.

Usage

To run extractFeatures, you first first navigate to the directory containing the executable inside of a UNIX/Linux terminal. There are two modes in which you can use the extractFeatures program. Mode 1 will extract the features of a single file and mode 2 will extract the features of a batch of files. The syntax to call the program are as follows:

Mode 1: **“**./extractFeatures filename”

In this mode features will be calculated for a single file. The filename argument should consist of the name waveform audio file (.wav) without the .wav extension.. The name should also include an acronym for the bird which is to be processed (more details can be found in the **Filenames** section). The features will be extracted into a comma separated value (CSV) file and its label will be output into a CSV as well. The features will be output into a CSV file which will contain the filename as well as “Training” and the label will be output into a CSV file which will contain the filename as well as “Label” in it . For example, to extract features from a file named “AC.wav” (AC standing for American Crow), “./extractFeatures AC” (assuming you are in the correct directory) will extract the features into a file named “ACTraining.csv” and its corresponding label into a file named “ACLabel.csv”.

Mode 2: “./extractFeatures folder/filename numFiles”

In this mode features will be calculated for a number of files with similar filenames. The folder/filename is used to denote the location and name of the files whose features are to be extracted. The filename convention is very strict and is outlined in the **Filenames** section. The numFiles argument denotes the number of files that are to be processed. The features and labels will be output into “TrainingData.csv” and “TrainingLabels.csv” respectively. For example, to extract features on a folder named “birds” which contain Northern Cardinal WAV files named “NC1.wav” through “NC30.wav” use the following command: “./extractFeatures birds/NC 30” and the features will be appended to “TrainingData.csv” and the corresponding labels will be output to “TrainingLabels.csv”, if the file did not exist prior to the execution of the program a new file will be created.

Note: In both modes the output CSV files are not overwritten, new data is appended to them as new lines (if the filename matches). If no output file exists by that name a new one is created.

Filenames

For the simplicity the program looks for certain acronyms to correctly label input files so that for the neural network if it is to be used for training. The current acronyms are as follows: "AC" for the American Crow, "AK" for the American Kestrel, "AYW" for the American Yellow Warbler, "BJ" for the Blue Jay, "CG" for the Canada Goose, "GWT" for the Green Winged Teal, and "NC" for the Northern Cardinal. For added birds please refer to the documentation. It is also important that input files do not contain the acronyms which refer to other birds, therefore all acronyms which are recognized by the system should be considered as reserved. For mode 2, the filename also includes the folder in which the files are located. Neither the folder nor the filename can have spaces in them. The files must have the same name and be consecutively numbered starting at 1. For example, to run extract features on a folder containing American Crow samples which is named "AmericanCrows" and the sample files within are named "AC1.wav", "AC2.wav", "AC3.wav", all the way to "AC50.wav", the following command can be used (assuming the folder is in the same directory as the extractFeatures executable): "./extractFeatures AmericanCrows/AC 50". The acronyms outlined still apply for mode 2 although in this mode is it not necessary for the files themselves to contain the acronym if the folder does, i.e. "./extractFeatures AC/AmericanCrow 50" would work if the folder name was "AC" and the files were consecutively named "AmericanCrow1.wav", "AmericanCrow2.wav" and so on to "AmericanCrow50.wav". It is also important that there are no gaps between the file numbers and that there are always files 1 through number numFiles. If the sample data is named differently or has gaps, consider either using mode 1 or writing a script (using a scripting language such as Python) to quickly rename the files.

Errors and Troubleshooting

If an error occurs please reread the user manual and double check that the program is being executed using the correct command. Also please reread the **Filenames** section as the format is very strict.

Neural Network

Command Line Arguments

```
BirdChirp_NN.py [-h] [-w WEIGHTS] [-e EPOCH] [-s SEED] [-l LEARN] [-n NODES]
        [-o OUTPUT] [-sp SPLIT] [-a ACCURACY] [-t SPLITBY] [-d TRAINALL] data labels
```

Positional Arguments

Name	Description	Type
data	csv file containing data	String
labels	csv file containing corresponding data labels	String

Optional Arguments

Flags	Name	Description	Type
-h, --help		show help message and exit	
-w , --weights	**WEIGHTS**	csv file containing weights	String
-e, --epoch	**EPOCH**	The number of training epochs	Positive Integer > 0
-s, --seed	**SEED**	Seed variable	Positive Integer > 0
-l, --learn	**LEARN**	Learning Rate	Positive Float > 0
-n, --nodes	**NODES**	Nodes in each hidden layer	Positive Integer > 0
-a, --accuracy	**ACCURACY**	Print test accuracy during each epoch	Boolean
-o, --output	**OUTPUT**	Name of output csv file containing weights	String
-sp, --split	**SPLIT**	Number of data separated from original data set	Positive Integer > 0
-t, --splitby	**SPLITBY**	Return split data as test data	Boolean
-d, --trainall	**TRAINALL**	Use all data as training data	Boolean

Default Parameters

EPOCH	SEED	LEARN	NODES	ACCURACY	OUTPUT	SPLIT	SPLITBY	TRAINALL
50000	100	.0002	150	False	weights	15	False	False

How to Train/Run the Model

- In the terminal, navigate to the directory containing the BirdChirp_NN.py file.
- Add the CSV files containing the data and corresponding labels to the same directory.
- Add the CSV file containing the list of bird names to the same directory.

Training the Model

Enter the following into the terminal and press ENTER:

- $ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv

Run the Model with preexisting weights

- Add the CSV file containing the weights to the same directory.

Enter the following into the terminal and press ENTER:

- $ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv -w (Name of file containing weights).csv

Using Different Parameters

EPOCH

The number of epochs is the number of times all of the training vectors are used once to update the weights. It is important to have large enough number of epochs to allow the model to learn the appropriate weights without underfitting however too large of number will overfit the model.

Usage:

- $ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv -e (Positive Integer > 0)

SEED

The seed changes which data samples become training data or testing data.

Usage:

- $ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv -s (Positive Integer > 0)

LEARNING RATE

Learning rate is how quickly the model will learn the optimal weights. However, if the learning rate is too high then the model might never be able to adjust the weights correctly. On the other hand, too low of a learning rate could take too long to properly adjust the weights.

Usage:

- $ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv -l (Positive Float > 0)

NODE

The number of hidden nodes in the hidden layer of the neural network.

Usage:

- $ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv -n (Positive Integer > 0)

ACCURACY

Setting the accuracy flag will allow the user to display the accuracy of the training data at each epoch to the console.

Usage:

- $ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv -a

Example:

```
$ python BirdChirp_NN.py Example_Data.csv Example_Labels.csv -a
```

OUTPUT

Every time the model is trained it will create a new CSV file in the same directory which contains all the weights. This file can later be used for predictions. If this parameter isn't used when training the model then the output file will use the default name "weights".

Usage:

- $ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv -o (Desired OUTPUT Name)

SPLIT

Split is the number of data samples that will be pulled from each species as training data and the remaining will be used for testing data.

Usage:

- `$ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv  -sp (Positive Integer > 0)`

Example:

```
$  python BirdChirp_NN.py Example_Data.csv Example_Labels.csv -sp 15
```

Potential Error:

Using a split that is greater than or equal to the number of samples of the bird with the lowest number of samples in the data set will result in an error.

SPLITBY

Splitby changes Split to be the number of data samples that will be pulled from each species as testing data and the remaining will be used for training data.

Usage:

- `$ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv  -t`

Example:

```
$ python BirdChirp_NN.py Example_Data.csv Example_Labels.csv -t
```

TRAINALL

Setting the trainall flag will negate the data splitting into separate training and testing data sets and will instead use all the given data as training data.

Usage:

- `$ python BirdChirp_NN.py (Name of file containing data).csv (Name of file containing labels).csv  -d`

Example:

```
$ python BirdChirp_NN.py Example_Data.csv Example_Labels.csv –d
```